DRAFT

Remarks

In the parent application, claims 1 to 16 and 18 to 21 were rejected. Applicant has cancelled claims 1 to 16 and 18 to 21, and added claims 22 to 48.

Drawings

Fig. 4 has been amended to replace reference numeral "88" with "88A" to avoid the use of the same reference numeral for two different elements.

Prosecution history of the parent application

In the June 20, 2000 Final Office Action of the parent application, the Examiner rejected claims 1 to 16 and 18 to 21 under 35 U.S.C. §112, second paragraph as being (1) indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention, (2) incomplete for omitting essential steps, and (3) incomplete for omitting essential structural cooperative relationships of elements. Applicant has cancelled claims 1 to 16 and 18 to 21, thereby rendering these rejections moot.

The Examiner also rejected claims 1 to 16 and 18 to 21 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Nos. 5,956,487 ("Venkatraman et al."), 5,982,445 ("Eyer et al."), 5,991,795 ("Howard et al."), and 6,011,537 ("Slotznick"), and "User Interface Tcchnologies for Home Appliances and Networks" by Peter M. Corcoran et al. ("Corcoran et al."). Applicant has cancelled claims 1 to 16 and 18 to 21, thereby rendering these rejections moot.

The Examiner further rejected claims 1 to 16 and 18 to 21 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,805,442 ("Crater et al.") in view of Official Notice and Applicant's own admission. Applicant has cancelled claims 1 to 16 and 18 to 21, thereby rendering these rejections moot.

New claims

Applicant has drafted new claims 22 to 48 in view of the prior §§112, 102(e), and 103(a) rejections. Applicant believes that claims 22 to 48 are patentable for at least the following reasons.

New apparatus claim 22

New claim 22 recites a GUI controller that frees an emb dded controller from handling a graphic user interface (GUI) for a device embedded with the two processors. As claim 22 is long,

DRAFT

Applicant has summarized the claim as follows. The GUI controller includes at least one memory that stores a document describing the GUI. The document does not include any executable codes but only operation codes (opcodes) that identify GUI objects and their parameters. The GUI controller also includes a processor that renders the GUI objects according their executable codes in a GUI object library stored in the at least one memory. The GUI objects are displayed on a liquid crystal display (LCD). When the GUI controller receives a command from a user through a touch screen, the GUI controller sends the command to the embedded controller and renders a first GUI objects accordingly. When the GUI controller receives a status from the embedded controller, the GUI controller renders a second GUI objects accordingly.

Venkatraman et al. does not disclose a GUI controller in addition to an embedded controller in a device. Venkatraman et al. teaches against the use of a separate GUI controller for handling the GUI of the device due to the added cost involved.

> The user-interface functions of such a device may be enhanced by the implementation of a screen-based user interface mechanism within the device. For example, such a device may include a display screen, and a rendering processor along with appropriate software for generating a rich graphical user interface suitable for the particular type of device. However, such screen displays and rendering mechanisms are usually expensive and increase the overall cost of the device. Such high costs are typically unsuitable for lower cost devices targeted for a relatively large mass market. Moreover, display screens and associated hardware may be too bulky for the size constraints of many devices.

Venkatraman et al., col. 1, lines 40 to 52 (emphasis added). To save cost, Venkatraman et al. discloses a single processor for controlling/monitoring the device and for rendering the GUI of the device. Venkatraman et al. states:

> FIG. 1b is a hardware block diagram of the device 10. The device 10 includes a processor 200, a memory 210, a set of device-specific hardware 300 along with a set of input/output circuitry 220 that enables communication via the communication path 22. The processor 200 performs device-specific functions for the device 10 in combination with the device-specific hardware 300. The processor 200 is also employed to provide web server functionality in the device 10. In one embodiment, the processor 200 stores the web page 18 in the memory 210 which may also be used to store information associated with normal device-specific functions.
>
> ....
>
> The web server functionality for the device 10 includes software executed by the processor 200 that services the HTTP protocol and that generates HTML formatted files. The web page 18 in one embodiment is stored in the memory 210 or may be

DRAFT

> generated on the fly. The processor 200 also executes communication software that drives the input/output circuitry 220 and provides the functionality of the network interface 12. In addition, the processor 200 executes software that performs control and information monitoring and logging functions of the monitor 16.

Venkatraman et al., col. 4, lines 5 to 16; lines 51 to 60 (emphasis added). Venkatraman et al. does not disclose a GUI controller in addition to an embedded controller in a device.

Howard et al. does not disclose a GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI. Howard et al. discloses a web browser 101 that communicates with an embedded device 102 through a proxy gateway 103.

> For a typical World Wide Web browser acting as a requesting communication unit 101, an embedded system as the responding communication unit 102, and a proxy gateway as the intervening communication unit 103, the document retrieval process may be in accordance with the following:
>
> First, the web browser initiates an HTTP GET request. The gateway receives the request, reformats it into enhanced HTTP, and forwards it to the device. The device sends the compressed document. The gateway then decompresses the document, and expands the compressed representation format.
>
> Next, the browser reads the document and does a GET for any needed documents, such as Java class files, that were referenced in the document. The gateway receives the GET request, but looks up and sends the file from its local storage.
>
> A compressed representation format may be placed in a document which is stored in an embedded device. When the document is retrieved from the embedded device, the communications gateway looks for a sequence that identifies the compressed representation format (escape sequence), then reads the group number following the escape sequence.
>
> The group number specifies what type of information element the sequence represents. Using the format specified by the specific group, the compressed representation format is decoded, then expanded into its full textual representation. This full (dynamically expanded) representation is then placed into the document by the gateway before forwarding the document to the software module which originated the document retrieval (the requesting communication unit).

Howard et al., col. 12, lines 29 to 58. As described above, embedded device 102 does not render a GUI. Embedded device 102 only sends a compressed document to web browser 101 upon request. Proxy gateway 103 intercepts the compressed document and expands the compressed document to a format recognized by web browser 101. Web browser 101 then renders the

DRAFT

uncompressed documents. As embedded device 102 does not render a GUI, it cannot disclose a GUI controller and an embedded controller that form a control system for a device embedded with these two controllers.

Eyer et al. does not disclose one GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI. Instead, Eyer et al. discloses a command processor 230 that receives user commands and a separate HTML/HTVP processor 215 that renders a display.

> The memory manager 225 receives commands from a user command processor 230, which, in turn, receives a user command via terminal 232. The user command may be input, for example, by a mouse or other pointing device, a keyboard, or an infrared remote control or the like. .... The memory manager 225 also receives commands from an HTVP processor 215.
>
> The HTML/HTVP processor 215 ... receives the HTML/HTVP data from the demultiplexer 205. .... The HTML/HTVP processor also receives commands from the user command processor 230. HTML/HTVP display data provided by the processor 215 is then provided to the combiner 250, where it may optionally be combined with the video data from processor 240 to produce a graphical display on a television screen wherein the display data overlays the programming service data, or vice-versa. ....

Eyer et al., col. 9, lines 1 to 31 (emphasis added). See also Eyer et al., col. 9, lines 46 to 50 ("when the user command processor 230 receives the user's command, it passes it to the HTVP processor 215 to cause the processor to provide the appropriate display data to the television"); col. 10, lines 64 to 67 ("user command processor 230 of Fig. 2 will be responsive to the user's command in sending control signal to the HTML/HTVP processor 215, memory manager 225, and/or the combiner 250 for controlling the output signal to the television"). Thus, Eyer et al. does not disclose one GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI.

Slotznick does not disclose a GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI. Instead, Slotznick discloses a method to reduce wait time for viewing web pages.

> A display controller causes the primary information to be displayed simultaneously with a portion of the secondary information on the user's display. When the user requests retrieval of subsequent primary information, a full display of the secondary information replaces the primary information in at least a portion of the delay time which occurs during retrieval of the subsequent primary information. The subsequently

DRAFT

> requested information is displayed after receipt thereof. The full display of secondary information is shown for a predetermined period of time, or may be held on the display by a user command. The user may also directly request a display of the full secondary information without requiring retrieval of subsequent primary information. The system may be implemented in an Internet environment wherein the primary and secondary information are retrieved from one or more remote websites. The portion of secondary information displayed which is simultaneously displayed with the primary information may be a thumbnail, keyhole or banner image of the full secondary information. The secondary information may be static, dynamic or user interactive.

Slotznick, abstract, lines 12 to 32. Thus, Slotznick discloses a computer system that downloads a web page including a full view of a primary information and a partial view of a secondary information. While the primary information is being viewed, the remainder of the secondary information is downloaded to reduce wait time. When the viewer decides to proceed, a full view of the secondary information is shown while the next primary information is downloaded to reduce wait time. Slotznick does not disclose a GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI. At most, Slotznick discloses a conventional computer where a display controller 212 is coupled a display 204 but not a user interface 210.

Corcoran et al. does not disclose a GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI. Corcoran et al. only discusses the merits of three user-interface technologies (i.e., personal/embedded Java, hand-held device markup language, and remote frame buffer) for remotely accessing consumer electronics in a home network or a wide-area-network.

Crater et al. does not disclose a GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI. Crater et al. discloses a controller 10 having a CPU 12 that operates a device and serves a web page containing device data to a querying computer 50.

> .... The controller 10 executes program instructions to operate, for example, a piece of industrial equipment, and includes a central processing unit ("CPU") 12 and one or more computer storage devices indicated generally at 14. ....
>
> CPU 12 and computer storage 14 communicate over an internal system bus 16. The system further includes a series of input/output modules shown representatively at 20_1, 20_2 that sense the condition of, and send control signals to, the controlled machine over a communication link (indicated by arrows). ....

DRAFT

> Storage 14 contains a series of functional blocks or modules that implement the functions performed by controller 10 through operation of CPU 12. A control block 35 contains instructions for operating I/O modules 20. These instructions are read in rapid sequence and interpreted to examine the condition of selected sensing devices associated with the controlled equipment, and, based thereon, to cause the controller to send appropriate operative control signals to the equipment. ....
>
> A network communication block provides programming to operate local-area network hardware and/or to connect with a remote network or network host. In the latter case, communication module 37 drives a modem within network interface 30 or other data-transmission circuitry to transfer streams of digitally encoded data over telephone or other communication lines.
>
> Storage 14 also contains data structures defining one or more web pages shown representatively at 40_1, 40_2. The web pages 40 consist of ASCII data obtained from one or more of I/O modules 20, HTML formatting instructions and associated data, and/or "applet" instructions that cause a properly equipped remote computer to display the data in a dynamic fashion. For example, an applet might cause temperature data to be displayed as a graphical representation of a thermometer, with the height of the rendered mercury column dynamically varying in proportion to the data from I/O modules 20 (and constantly provided to the remote computer via network interface 30); pressure data might be represented in the form of a graphically rendered needle-type pressure gauge. .... Management and transmission of web pages 40 to a querying computer is handled by a web server module 45, which allows controller 10 to function as a network server, ...
>
> Incoming data from I/O modules 20 may be processed by control block 35 before being copied into one of the web pages 40. Because of the linking capabilities of the web, it is not necessary for the data to be stored in the web page containing the display instructions; instead, the latter page may contain a "hyperlink" pointer to a different web page in which data is accumulated. In addition, a web page can obtain data from other web pages (e.g., from different controllers) by accessing those web pages when appropriate. For example, if a cluster of controllers is operationally related such that data from one is usefully combined with data from the others, each page of the cluster can contain instructions to access the other pages (or their associated data pages) when accessed by a user, and the applet configured to present data from the entire cluster. Alternatively, the applet can be configured to cause the client's browser to access the web page. As used herein, data is "associated with" a web page or an applet if it is stored as part of the web page or applet, or stored in a directly or indirectly hyperlinked web page.

Carter et al., col. 6, line 21 to col. 8, line 3 (emphasis added). Thus, Crater et al. teaches a CPU 12 that acts both to control an industrial equipment and to generate web pages containing data collected from the industrial equipment. Crater et al. does not disclose a

DRAFT

GUI controller that is coupled to a touch screen to receive a user command and to an LCD to display the GUI.

New apparatus claims 23 to 34

New claim 23 recites a first controller (e.g., a GUI controller) that provides a GUI for a device that is monitored and controlled by a second controller (e.g., an embedded controller). Specifically, the first controller includes at least one memory that stores a document describing the GUI. The document does not include any executable codes but only opcodes that identify a GUI object and its parameter. The first controller further includes a processor that renders the GUI and a nonvolatile memory that stores the executable codes for the GUI object. Carrying out the executable codes, the processor communicates the parameter with its source and renders the GUI object in response to the parameter. The references discussed above do not disclose these elements. Accordingly, Applicant believes that claim 23 is patentable.

New claims 24 to 34 depend from claim 23 and are believed to be patentable for at least the same reasons that claim 23 is patentable.

New method claim 35 to 48

Applicant believes that new method claims 35 to 48 are patentable because they are directed to subject matters similar to claim 23.

Information Disclosure Statement

Along with this Preliminary Amendment, Applicant submits an Information Disclosure Statement and a USPTO Form 1449 citing articles reviewing the "Easy GUI Controller Chip," which represents one embodiment of the claimed invention. These reviews recognize the novelty and the benefits of the claimed invention. One article states:

> Amulet's browser chip serves as both LCD controller and user interface engine. The intended architecture has the main embedded application running on a separate processor, communicating with Easy GUI via an RS-232 serial interface. This arrangement significantly boosts efficiency by freeing the main microprocessor and memory of any graphics work. An additional benefit is that much of the user interface logic and all of the text and images can be revised or upgraded with no changes to the main embedded device firmware.
>
> ....
>
> Interfaces for the Easy GUI are laid out in the same manner as an HTML website, a strategy clever in several ways. The standard and

DRAFT

universal nature of HTML allows GUI prototype screens to be viewed in any web browser. Such accessibility makes the demonstration and testing of new interfaces as simple as visiting a web page. And advanced HTML editors like Macromedia Dreamweaver and Adobe GoLive can now be used to perform LCD interface design.

Using tools included on a CD ROM, I successfully built and deployed a basic touch screen interface. Text, buttons, and images built up quickly as I followed useful examples in the MS Windows-based development kit. I compiled my new interface into native "μHTML" required by the display chip. Voila! In a single button click, my screens were up and running on the LCD display. The PC's standard 9 pin DIN "COM" port communicated over the supplied cable without a hitch.

The Easy GUI compiler is competent in handling GIF and JPEG monochrome images as well as most HTML presentation tags. To achieve "flipbook" style animation for screen components, animated GIF images are rendered in motion and include speed control. A suite of scaleable proprietary "widgets" provides interface elements lacking in HTML, such as bar graphs, line plots, and sliders.

....

Once an interface is compiled, uploaded, and running on the GUI controller, communication with the main process occurs over the serial port. Special codes embedded in the HTML invoke "widgets" to send and receive messages. Using this technique, a timer or user input from the touch panel initiates events. The Easy GUI client acts as master and can send five different types of messages including "get," "set," and a "remote procedure invoke." Included is a PC-based simulator allowing serial communications to be tested without a slave embedded device.

While lacking the ultimate flexibility of traditional code-driven interfaces, Amulet's innovative use of HTML makes up the difference with simplicity and rapid development. With a list price of $399, Easy GUI is a unique system that should reduce labor and save time, particularly on embedded projects with basic to moderate screen demands.
....

"Tool saves time in LCD-interface design" by Jude DeMeis, Design News, April 8, 2002 (emphasis added). Another article states:

One option when you want to manage your graphics from a serial port is the Amulet Technologies GUI. This product contains an LCD, a touchscreen, and a small board to drive the LCD. In addition to managing the data and clock signals required by the LCD, the board provides a graphical library and some simple graphical objects.

You don't program this board in C. Instead, you load an HTML file to be stored in flash over the serial port. The serial port can then provide a conduit for messages that change values on the display. Such a message might be a temperature value sent to update the height of a bar.

DRAFT

Messages received from the Amulet indicate when an ev nt, such as a user pressing a button, has occurred.

The downloaded HTML file is not pure HTML, but the variations are slight. They call it HTML. While the format is similar to HTML, the user's experience is not mediated by a browser. The HTML format is simply a processor-independent way to describe the layout of a screen or a number of screens and locate text, bitmaps, and graphical widgets on those screens.

If you've defined a number of screens, navigating among them doesn't require any communication external to the Amulet. A user event, such as a changed value on a slider, will trigger communication. This minimizes the frequency of interruptions to the normal work of the microcontroller. In the other direction, the processor may transmit new values to the Amulet to update values, text, or widgets on the display.

This design creates a useful division between the control processor and the graphics engine. Since the a [sic] HTML file is the medium for graphics control, the graphics board also doesn't require any code. You have to program the control processor to perform some serial communications, but you don't have to bother with line-drawing routines and the like. More conventional graphics solutions offer a software library that you integrate with your own program. Amulet provides the low-level graphics as a hardware solution.

If you change the layout or redesign some icons, you just disconnects [sic] the graphics module from the controlling processor and moves [sic] it to a serial port of the PC. A new HTML file downloads, and now you've got your new user interface is in place. The upshot? Purely visual changes don't require modifications to the microcontroller software.

Since the Amulet interacts with the rest of the system via serial protocol, integration with a specific processor, RTOS, or compiler is not an issue. For this reason, the Amulet is by far the fastest design solution I have seen for adding a GUI to an existing system. ....

"GUI add-on options" by Niall Murphy, Embedded Systems Programming, April 2, 2003 (emphasis added). These articles are provided to assist the Examiner in understanding the claimed invention.

In summary, Applicant has cancelled claims 1 to 21 and has added claims 22 to 48. Applicant respectfully requests entry of the above amendment. Should the Examiner have any questions, the Examiner is invited to call the undersigned at (408) 382-0480.

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner For Patents, Washington, D.C. 20231, on _____.

_____ _____
Attorney for Applicant(s) Date of Signature

Respectfully submitted,

DRAFT

David C. Hsia
Attorney for Applicant(s)
Reg. No. 46,235